Reliable image notifications for smart home security with MQTT

Nazir, Sajid; Kaleem, Muhammad

Published in:
2019 International Conference on Information Science and Communication Technology, ICISCT 2019

DOI:
10.1109/CISCT.2019.8777403

Publication date:
2019

Document Version
Author accepted manuscript

Reliable Image Notifications for Smart Home Security with MQTT

Sajid Nazir[1], Muhammad Kaleem[2]
sajid.nazir@gcu.ac.uk
[1]*School of Computing, Engineering and Built Environment, Glasgow Caledonian University*, Glasgow, G4 0BA, UK
[2]*Department of Engineering, COMSATS Institute of Information Technology*, Islamabad, Pakistan

***Abstract*— Internet of Things (IoT) applications utilize low-powered battery operated devices and reliable event notifications depend on low-power messages from the sensors node to the server and vice versa over bandwidth constrained unreliable channels. Publish/subscribe communications protocols play an important part in such low-powered device communications with MQTT protocol being most prevalent for such deployments. These protocols have enabled IoT applications such as smart home, fleet management, oil pipelines and space exploration that heavily depend on reliable communications. This paper investigates utilization of MQTT protocol for a smart home security system. A Raspberry Pi implementation detects an intrusion event using a passive infrared sensor triggering an image capture which is then communicated to the subscribed client for countermeasures. Results for power consumption and data transfer are provided that help understand the behaviour with different service qualities.**

Keywords—Notifications; Internet of Things; M2M; home security; power measurements; image communications;

I. INTRODUCTION

The usage of sensors for monitoring and control in Internet of Things (IoT) applications is increasing and relies heavily on timely and reliable event notifications for efficient operation. The IoT device deployment is projected to cross 30 billion by 2020 [1] and more application domains are embracing the sensing technologies in novel ways to reap the benefits of off-site reliable and low cost monitoring. Polling is one way to detect events but does not work for large scale of devices and time constraints of critical applications.

In publish/subscribe notification model, the producers and consumers of messages are decoupled through an intermediary or message broker that takes over responsibility for message delivery. A message producer or client publishes to a topic with a broker, which pushes notifications down to all consumer clients who have subscribed an interest in receiving information on that topic [2]. Thus the publisher and subscriber devices economize on power with the message delivery delegated to the server. In terms of placement, a message broker or server could be deployed as private server or could be hosted on public cloud such as Google Messaging Service (GMS).

These low power communications protocols are suitable for power constrained Systems on Chips (SoC) devices by eliminating the need for polling and still ensuring reliable delivery. There are many popular M2M protocols like Message Queuing Telemetry Transport (MQTT) [2], Constrained Application Protocol (CoAP) [3] and Open Mobile Alliance Light Weight M2M (OWA LWM2M). These and other protocols for IoT applications have been compared [4][5] concluding that although each protocol has its own merits and suits different communications, however, MQTT is better suited for event-based applications with its publish/subscribe model. It also provides different modes of delivery, low power consumption and security.

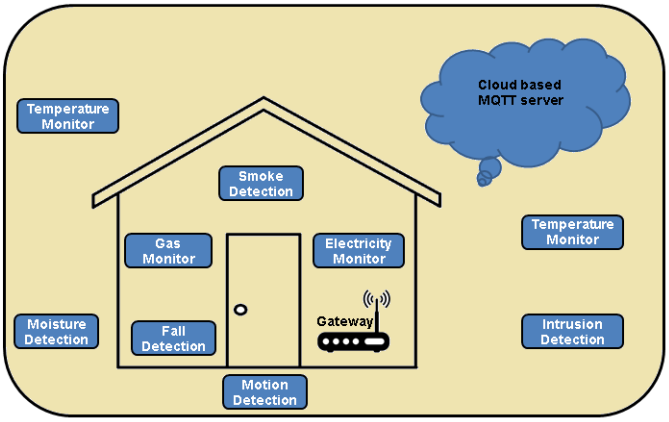

Fig. 1. A general IoT Smart Home Architecture for event notification system.

MQTT is a client server based notification protocol (thus decoupling the producers and consumers) model particularly suitable for the low-power devices. It was initially designed for Supervisory Control and Data Acquisition (SCADA) systems for oil pipelines and satellite links [6] and was later commercially developed by IBM [7]. It was designed for messaging over TCP with low-bandwidth, high latency and unreliable networks in mind. A real-world use includes Facebook messenger app which was based on this protocol to suit battery powered mobile devices [8].

A smart home has many sensors and actuators deployed both within and outside the home (Figure 1) exchanging notifications to effect efficient operations in, for example, energy utilization. Other than these scalar measurements, such as temperature, security can be assured through timely detection and notification of a physical intrusion through

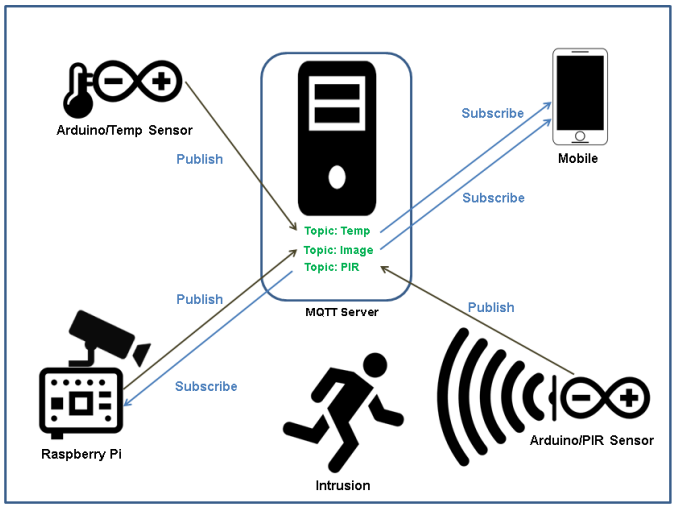

Fig. 2. MQTT Clients exchanging event information in a security system.

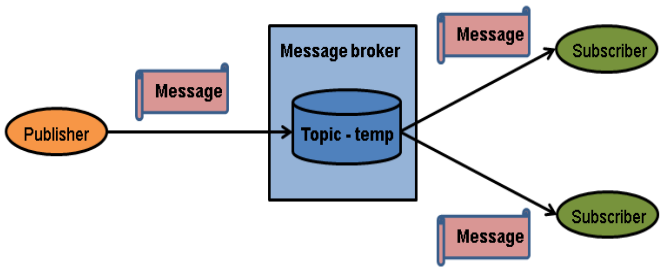

Fig. 3. Client server interactions.

camera nodes capturing image events. Besides an image also provides rich contextual information for other events such as fire, falls etc. Figure 2 shows MQTT clients exchanging messages through MQTT server.

In this paper we propose a security system for a smart home based on MQTT protocol to capture and transmit images for intrusion investigation. This paper has the following contributions: (1) presents a case study for home security using MQTT for image event notifications, and (2) provides the power measurements and packet transfers for image transmissions.

Rest of this paper is structured as follows: Related work is described in Section II. The relevant details of MQTT protocol are covered in Section III. An event notification system based on MQTT deployment for a smart home is described in Section IV. The results and discussion are provided in Section V. Finally, Section VI concludes the paper.

II. Related Work

There have been numerous studies employing the study of MQTT protocol for event notifications for diverse and challenging environments. The uses for health monitoring [9], security [10] and agriculture area [11] are just a few of example areas. An implementation of MQTT for monitoring of remote units, into a distribution network is described in [5] improving the quality of service in voltage grids. A simplified

TABLE I. MQTT Packet Header Format [2]

Bit	7	6	5	4	3	2	1	0
byte 1	Message Type				DUP Flag	QoS level		RETAIN
byte 2	Remaining Length							

infrared heart rate monitor transmitting to MQTT server housed on a Raspberry Pi is proposed in [12] that offers many advantages over HTTP and provides a low cost measurement for remote locations. Latency comparisons of AMQP and MQTT [13] show that the MQTT provides better average latency.

MQTT was used to devise a scalable energy management system to investigate HVAC system [14]. Transformation of a traditional to smart home through design and implementation of a reliable and economical sensor network using MQTT and openHAB is proposed in [15]. The basic idea was to utilize wireless nodes as replacement for switch boards. The use of control protocols for pedagogy is proposed in [16][17].

A smart home system based on MQTT broker housed on Amazon Web Services (AWS) to control room temperature, alarm sensing, and suppress fire is described in [18].

The power consumption of MQTT protocol and the effect of selecting various QoS are described in [19]. The power consumption was recorded on the Raspberry Pi and Arduino using a USB Drok digital multimeter. A home automation system is proposed in [20].

III. MQTT Protocol

MQTT protocol is defined on top of Transport Control Protocol (TCP) [2]. A variant of MQTT protocol, MQTT-SN (sensor networks) [21] is defined for wireless communication environments with high link failures and low bandwidth such as ZigBee.

The central idea to share the notifications is that of topics. A client or the server can publish topics of interest to which a client can subscribe. Figure 3 shows the message communication to the subscribers. The major benefit of using a subscription based protocol is that receiver node is guaranteed delivery of the message even in case of temporary disruption of the communication channel. Topics may be separated by '/' for structured topic hierarchies [6]. Mosquitto server is able to translate and transfer messages between MQTT and MQTT-SN, so that it can act as a gateway [22] between devices which communicate with either protocol.

A. Packet Header

The packet header shown in Table I consist of 2 bytes and packet sizes depend on the payload as not all packets contain payload. As shown in Table I, there are 15 message types defined. DUP (duplicate) is set when client or server attempts to re-deliver a message and the recipient should treat it as a hint [2]. The QoS level indicates the delivery guarantees enforced based on the significance of a message defined in detail in Section C. If the RETAIN flag is set in a PUBLISH message then the server holds on to the message after delivery to the current subscribers. With a new subscription, the last

TABLE II. MESSAGE EXCHANGES FOR DIFFERENT QOS [2]

Message	Quality of Service		
	Messages-QoS 0	**Messages-QoS1**	**Messages-QoS 2**
PUBLISH	Client->Server	Client->Server	Client->Server
PUBREC	-	-	Server->Client
PUBREL	-	-	Client->Server
PUBCOMP	-	-	Server->Client
PUBACK	-	Server->Client	-

retained message is sent to the subscriber which has RETAIN flag as set. Byte 2 for remaining length indicates remaining bytes in the current message including variable header and payload. The variable header is placed in between fixed header and the payload [2]. The payload in MQTT packet can be from 0 to 256 MB. However the protocol is agnostic to the data contents.

B. Keep-alive messages

The TCP connection between MQTT subscriber client and server is kept open in normal operation and the subscriber client must send a message within Keep Alive interval or in the absence of a data message client sends a PINGREQ message which is acknowledged by the server with a PINGRESP. If the server does not receive a message from the client within one and a half times Keep Alive value then it disconnects the client. In case a PINGRESP is not received within Keep Alive interval then the client should close the TCP/IP connection [2].

C. Quality of Service Levels

The MQTT protocol [2] specifies three modes of communication based on Quality of Service (QoS) to deliver a message to the subscribers.

1) QoS 0: At most once: This is like 'Fire and Forget', that is, the message is not acknowledged. However, loss and duplication can occur.

2) QoS 1: At least once: This ensures acknowledged delivery although duplicates could occur.

3) QoS 2: Exactly once: This ensures assured delivery of the message to the subscribers but exactly once.

4) QoS 3: Reserved.

D. Security

The MQTT protocol provides Transport Layer Security (TLS) and the ability to secure communications using Secure Socket Layer (SSL)/TLS. In addition, the Mosquitto server has the ability to restrict user access to MQTT topics.

IV. SMART HOME SECURITY EVENT NOTIFICATION

The smart home event notification system is an implementation of an event based notification system on the home network based on Raspberry Pi (a Linux based computer) [23] based Mosquitto server. The system monitors the entrance (and additionally other such interest areas) and can send a notification to the subscribed devices both within and outside the smart home.

We host Mosquitto server, which is an open source project on Raspberry Pi which can be interfaced with different types of sensors and actuators using its General Purpose Input Output (GPIO) pins and other interfaces. In a smart home environment it can be interfaced with different modalities of interest. The major benefit of its use are that it can be connected to the home network and send/receive messages and its on-board storage can be tailored to suit an application.

The MQTT client that used for Android platform is MyMQTT app freely downloadable from the Google App store. Paho MQTT libraries were used on Raspberry Pi for writing Python scripts.

A. System Configuration

The system configuration is shown in Figure 2. It consists of a Raspberry Pi hosting MQTT broker/server. There are four MQTT clients, one hosted on Raspberry Pi with a camera, second is Arduino with a temperature sensor and the third is an Arduino with a PIR sensor. The mobile client is Samsung Galaxy S8. All the clients and servers are connected to the home network.

1) Message Broker: Raspberry Pi was used as a processing unit that was connected to the home network. The MQTT Mosquitto server is placed on the Raspberry Pi.

2) Event Registration: The events are based on motion activated image captures. In addition, temperature was used for another event notification.

B. Operation

The operation is explained in the following sub-section.

1) Data Capture: The data was captured through the sensors directly connected to the Raspberry Pi. We utilise both scalar sensors such as temperature sensor and a camera as a sensor that can identify motion events in an area of interest.

2) Event Generation: There are two types of events generated in the system. It could be based on motion detection through a PIR sensor and then sending the captured images to all subscribers. The other event is the temperature measurement on the Arduino through an attached temperature sensor.

3) Event Notification: The events are notified to all the interested subscribers that have registered an interest through a subscription.

C. Packet sniffing

The MQTT packets were captured using Wireshark [24] on a Windows laptop connected to the home network. Two types of measurements are of an interest. Firstly during a silent period PINGREQ and PINGRESP exchanged between the server and the client can be captured. Secondly, in case of an image transmission, the data packets are captured to determine packet sizes.

D. Power Consumption

In order to measure the power consumption for an image transmission we used a Drok USB tester connected to a Raspberry Pi. For better accuracy we transmitted a 640x480 resolution image repeatedly for 10 times and obtained the average power measurement.

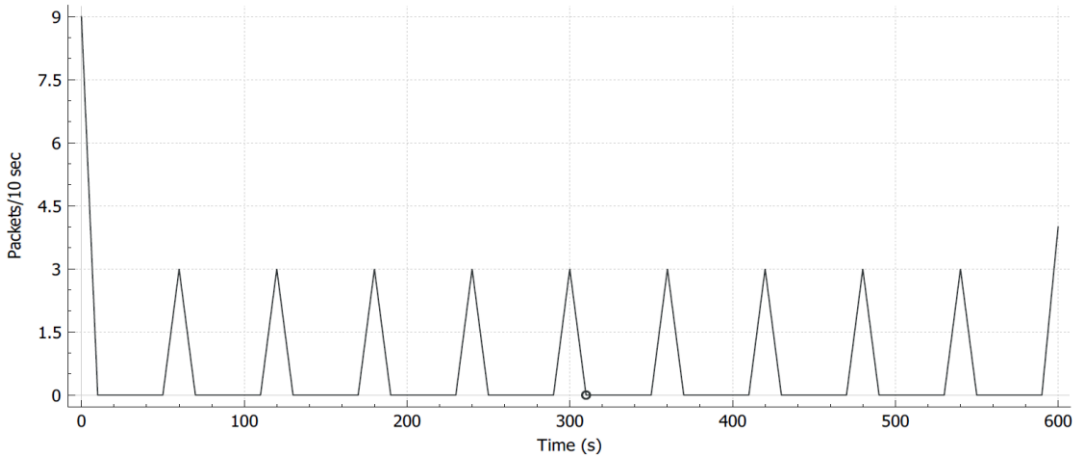

Fig. 4. Communication between MQTT server and subscriber. Connection setup, PINGREQ, PINGRESP and connection teardown for 10 minute duration over port 1883.

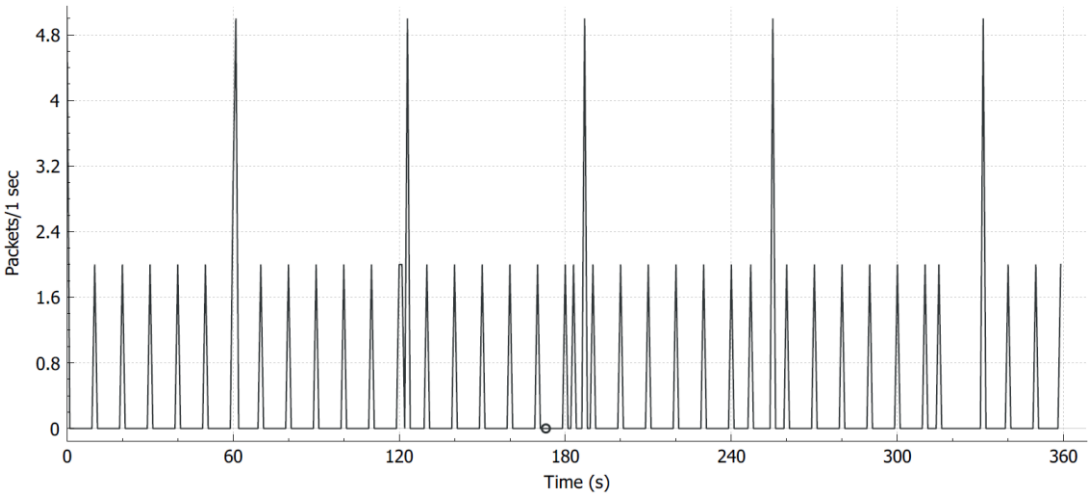

Fig. 5. Communication between MQTT server and a Publisher publishing every 10 Seconds to a topic captured over a 6 minute period. A connection request is being sent every 60 seconds.

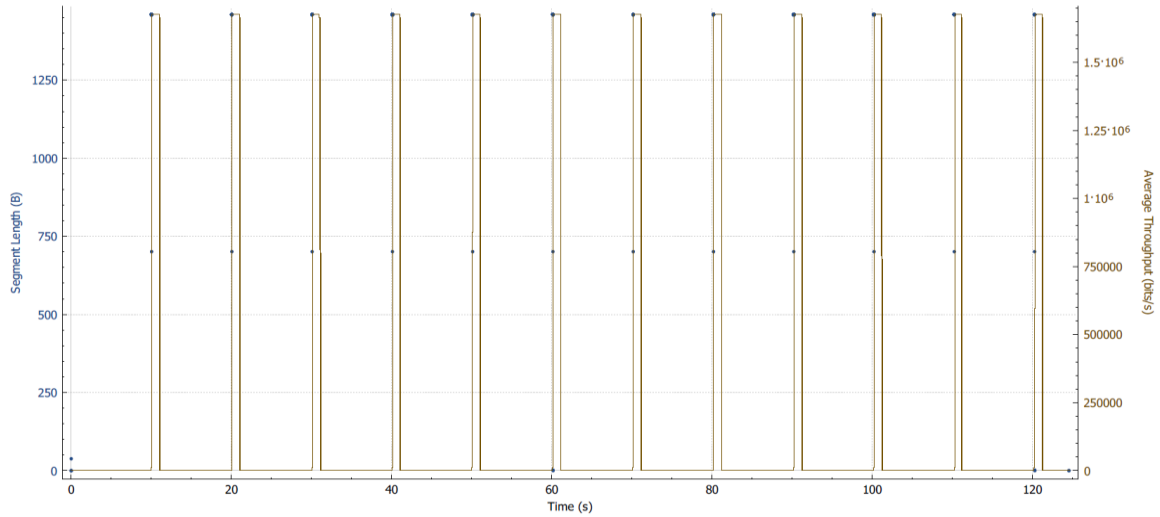

Fig. 6. Throughput for a 209469 byte image transfer from a MQTT publisher to a MQTT subscriber through a python script every 10 seconds over a 2 minute period.

V. Results and Discussion

A. Packet Transmission for Different QoS

The packet exchanges are minimum for QoS 0 and increase as the QoS increases to 2 as shown in Table II. The results for setting up a connection from a MQTT subscriber with MQTT server and the packet exchanges that take place for keeping the connection open are shown in Figure 4. The default Keep Alive setting of 60 seconds was used. Figure 5 shows communications between MQTT server and MQTT publisher publishing a small 16 bytes message every 10 seconds. A connection request can be seen occurring at 60 seconds interval.

B. Image Transmission

The image transmission from the Raspberry Pi sensor node to the MQTT server takes place in smaller packets where the fragmentation is due to the network MTU that is 1514 bytes.

The result for packet capture is shown in Figure 6. The graph shows transmissions of 12 images in total. Similar to Figure 5 there is a connection request being sent every 60 seconds (not discernible in the figure).

C. Power Consumption

The power consumption was measured on Raspberry Pi for image transmission. For normal operating condition, the voltage was measured as 5.3V, whereas the current draw was measured as 0.24A with an Ethernet connection and 0.35A with a USB WiFi dongle. The power measurements for an image transmission with WiFi were an additional 1.28 W for just 800 ms. The image data is much larger than the other MQTT packet exchanges for connection establishment and those for different QoS.

VI. Conclusion

Timely and reliable intrusion detection and event notification is critical for security applications. The proliferation of smart mobile devices thus enables a person to be continuously aware of the interesting events taking place at a remote location. MQTT is a low power protocol for event notification. This paper provides details of message exchanges for different quality of services between MQTT clients and subscribers.

MQTT protocol is useful to provide notifications in diverse applications such as home security, fall detection for elderly people, welfare of the pets, or any other modality of interest such as fire, leakage etc. Transmission of an image along with an event helps to establish a context for the first responders.

In our future work we plan to extend the basic system model discussed in this study by hosting the server in the cloud which will allow MQTT clients to publish/subscribe from anywhere in the world.

References

[1] A. Nordrum, "Popular Internet of Things Forecast of 50 Billion Devices by 2020 Is Outdated," *IEEE Spectrum*, Aug 2016.

[2] MQTT V3.1 Protocol Specification, International Business Machines Corporation (IBM) Eurotech.

[3] Z. Shelby, RFC-7959 Block-Wise Transfers in the Constrained Application Protocol (CoAP), Aug 2016.

[4] K. Fysarakis, I. Askoxylakis, O. Soultatos, I. Papaefstathiou, "Which IoT Protocol? Comparing standardized approaches over a common M2M application," *2016 IEEE Global Communications Conference (GLOBECOM)*, Washington, DC, 2016, pp. 1-7.

[5] H. Eslava, L. A. Rojas, R. Pereira, "Implementation of Machine-to-Machine Solutions Using MQTT Protocol in Internet of Things (IoT) Environment to Improve Automation Process for Electrical Distribution Substations in Colombia, *Journal of Power and Energy Engineering*, 2015, 3, 92-96.

[6] H. Kamutzki, "Using MQTT In Real-World M2M Communication," Talk on MicroDoc blog. Available online: https://www.microdoc.com/blog/using-mqtt-real-world-m2m-communication.

[7] J. Butts, "The Information Revolution for the Enterprise," IBM. https://opcfoundation.org/information-revolution-2014/

[8] L. Zhang, Building Facebook Messenger, Aug 2011. Available online: https://www.facebook.com/notes/facebook-engineering/building-facebook-messenger/10150259350998920

[9] A. Kaur, A. Jasuja, "Health Monitoring Based on IoT using Raspberry Pi," *International Conference on Computing, Communication and Automation (ICCCA2017)*.

[10] J. Prabaharan, A. Swamy, A. Sharma, K. N. Bharath, P. R. Mundra and K. J. Mohammed, "Wireless home automation and security system using MQTT protocol," *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, Bangalore, 2017, pp. 2043-2045.

[11] K. Grgić, I. Špeh and I. Heđi, "A web-based IoT solution for monitoring data using MQTT protocol," *2016 International Conference on Smart Systems and Technologies (SST)*, Osijek, 2016, pp. 249-253.

[12] K. Chooruanga, P. Mangkalakeeree, "Wireless Heart Rate Monitoring System Using MQTT," *2016 International Electrical Engineering Congress, iEECON2016*, 2-4 March 2016, Chiang Mai, Thailand.

[13] E. Linden, Master Thesis, "A latency Comparison of IoT protocols in MES", Linkoping University, 2017.

[14] A.R. Al-Ali, I. A. Zualkernan, M. Rashid, R. Gupta, M. AliKarar, "A smart home energy management system using IoT and big data analytics approach," *IEEE Transactions on Consumer Electronics*, Vol. 63, No. 4, November 2017.

[15] A. Bhatt, J. Patoliya, "Cost Effective Digitization of Home Appliances for Home Automation with low-power WiFi devices," *International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB16)*.

[16] M. A. Prada, P. Reguera, S. Alonso, A. Mor´an, J. Fuertes, M. Dom´inguez, "Communication with resource-constrained devices through MQTT for control education," *IFAC-PapersOnLine*, 49-6 (2016) 150–155.

[17] A. Oh, "Design and Implementation of MQTT based on Arduino," *Information*, vol. 21, no. 2, pp. 581-588.

[18] D. H. Kang et al., "Room Temperature Control and Fire Alarm/Suppression IoT Service Using MQTT on AWS," *2017 International Conference on Platform Technology and Service (PlatCon)*, Busan, 2017, pp. 1-5.

[19] A. Viswanathan, MSc thesis, "Analysis of Power Consumption of the MQTT Protocol", University of Pittsburgh, 2017.

[20] R. K. Kodali, S. Soratkal, "MQTT based Home Automation System Using ESP8266,"

[21] MQTT For Sensor Networks (MQTT-SN) Protocol Specification, Version 1.2

[22] S. Kim, H. Choi, W. Rhee, "IoT Home Gateway for Auto-Configuration and Management of MQTT Devices.," *IEEE Conference on Wireless Sensors*, 2015.

[23] Raspberry Pi: https://www.raspberrypi.org/about/

[24] Wireshark: https://www.wireshark.org/